

Available online at [www.sciencedirect.com](http://www.sciencedirect.com)

SCIENCE @ DIRECT®

Science of Computer Programming 53 (2004) 87–105

Science of  
Computer  
Programming[www.elsevier.com/locate/scico](http://www.elsevier.com/locate/scico)

## A Java-based digital library portal for geography education

Zehua Liu<sup>a</sup>, Hai Yu<sup>a</sup>, Ee-Peng Lim<sup>a,\*</sup>, Ming Yin<sup>a</sup>,  
Dion Hoe-Lian Goh<sup>b</sup>, Yin-Leng Theng<sup>b</sup>, Wee-Keong Ng<sup>a</sup>

<sup>a</sup>*Centre for Advanced Information Systems, School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore*

<sup>b</sup>*School of Communications and Information, Nanyang Technological University, Singapore 639798, Singapore*

Received 6 September 2003; received in revised form 2 February 2004; accepted 15 February 2004

Available online 9 June 2004

---

### Abstract

G-Portal is a Java-based digital library system for managing the metadata of geography related resources on the Web. In addition to providing a flexible repository subsystem to accommodate metadata of different formats using XML and XML Schemas, G-Portal organizes metadata into projects and layers, and supports an integrated and synchronized classification and map-based interfaces over the stored metadata. G-Portal also includes a classification subsystem that creates category structures and classifies metadata resources into categories based on user-specified classification schemas. Furthermore, G-Portal users can annotate resources and make their annotations available to others. In this paper, we describe the design and implementation of G-Portal and elaborate how Java is used to implement its features. G-Portal has been designed to be modular and some of the modules can be used as stand-alone tools. In this paper, we use UML notation to describe the detailed design of G-Portal and highlight some of the design decisions.

© 2004 Elsevier B.V. All rights reserved.

**Keywords:** Digital libraries; Geography education; Metadata management

---

---

\* Corresponding author.

*E-mail addresses:* [liuzh@pmail.ntu.edu.sg](mailto:liuzh@pmail.ntu.edu.sg) (Z. Liu), [ashyu@ntu.edu.sg](mailto:ashyu@ntu.edu.sg) (H. Yu), [aseplim@ntu.edu.sg](mailto:aseplim@ntu.edu.sg) (E.-P. Lim), [asmyin@ntu.edu.sg](mailto:asmyin@ntu.edu.sg) (M. Yin), [ashlgoh@ntu.edu.sg](mailto:ashlgoh@ntu.edu.sg) (D.H.-L. Goh), [tyltheng@ntu.edu.sg](mailto:tyltheng@ntu.edu.sg) (Y.-L. Theng), [awkng@ntu.edu.sg](mailto:awkng@ntu.edu.sg) (W.-K. Ng).

## 1. Introduction

### 1.1. Background

The ubiquity of World Wide Web has brought about significant changes to the role of libraries in schools, universities, public and private institutions. The Web provides an enormous amount of information in web pages that can be browsed for free. Compared to the situation a decade ago, students and researchers today are more likely to patronize the Web instead of visiting physical libraries for information.

In this paper, we introduce a digital library portal known as **G-Portal** that attempts to provide better selection, organization, and indexing of the Web content of educational values. More specifically, G-Portal is designed to manage the metadata of geography related Web resources which can be web pages or any other Web objects that may be relevant to the geography education in junior and senior high schools. Examples of such Web resources include a web page describing the mountains in East Malaysia, or a detailed map of China. Metadata here refers to data about data. Similar to library catalogues, G-Portal's metadata provide a description of the actual Web resources and the URL references to the latter. The similarity between G-Portal metadata and library catalogues however ends here as the referenced Web contents are found within the global Web and are therefore not within the control of G-Portal.

Moreover, unlike those of other Web portals, the design of G-Portal must address three important characteristics unique to its metadata content:

- **Spatial attributes.** Many of G-Portal's metadata can be associated with locations and shapes on a map. These are known as the **geospatial metadata**. For example, for web pages containing the census data for different states in USA, each state's census data web page can be associated with a polygon representing the state boundary. It is therefore necessary for G-Portal to support digital library services dealing with geospatial metadata.
- **Diverse metadata structure.** The broad range of geography related Web content suggests that there is no one single structure that can be used for representing G-Portal's metadata. This is very different from the case for usual Web portal and library catalogue data where a single set of attributes suffice.
- **Lack of a standard set of subject categories.** There is currently no standard set of subject categories defined for Web content. Hence, it is not feasible to expect metadata records created in G-Portal to be assigned subject categories similar to those of library category records.

Due to the above unique metadata characteristics, the design of G-Portal has incorporated several novel ideas into the digital library services provided to its users. These include:

- a flexible classification scheme for categorizing metadata resources;
- a versatile mechanism for logically organizing metadata resources;
- a resource visualization module that allows the same metadata resource to be navigated in both map-based and classification-based user interfaces;

- a query module that supports both spatial and non-spatial queries on metadata of different formats.

To allow users to easily access to the above features, we have chosen to implement G-Portal in Java. Apart from being object oriented, Java is ideal for implementing the graphical user interfaces required by G-Portal. For example, the viewing and manipulation of maps can be made much simpler if users can carry out zooming, panning, and other operations interactively with G-Portal. Furthermore, by implementing the G-Portal client as a Java applet, G-Portal can run on any Java-enabled Web browser without any installation efforts. In other words, G-Portal can become platform independent, making itself available to any user on the Web.

On the server side, the decision to also use Java to implement the server program is mainly for the ease of communication with the client, since the client is implemented using Java. Portability is another important consideration that affects our decision. In fact, the project has taken great advantage of the platform-independent feature of Java by having the development on a Windows platform and the deployment on a Sun Solaris server. In addition, Java is favoured because of its faster application development time due to the wide availability of high level API and third-party packages.

### *1.2. Objectives*

In this paper, we will describe the design and implementation of G-Portal. We will place much emphasis on some interesting ideas realized in G-Portal making it a unique digital library portal. The objective here is to share our G-Portal experience with the researchers and developers of other similar applications.

At the system level, G-Portal consists of several essential functional modules which are designed to be reusable for other similar applications. EXtensible Markup Language (XML) and other XML facilities also have been used extensively in G-Portal. At the component level, the Java packages and classes for realizing the functional modules will be described. We will also outline the implemented methods.

While G-Portal can be viewed as a learning tool for users wishing to acquire geography related knowledge, its current design is being extended to support e-learning activities. Nevertheless, due to the rather preliminary findings and space constraints, we will exclude the e-learning aspects of G-Portal from this paper. Interested readers can refer to [3] for more detailed information about the user study on G-Portal involving high school students and geography examination resources.

### *1.3. Related work*

There have been several ongoing research efforts making Web content available to users in a more structured and organized way [1,2,4,5,7,8,12–16,18]. In this section, we give a brief review of those that are most closely related to our G-Portal project.

The Alexandria Digital Project (ADEPT) [15] is a project initiated by the University of California, Santa Barbara, to host data related to earth sciences. The ADEPT research focuses on the creation and management of data content. The research further supports the maintenance of metadata, their organization, and spatial (and non-spatial) query processing

on them. The ADEPT prototype system, however, differs from G-Portal by not adopting flexible metadata representations and classification. In fact, ADEPT assumes that all its data are hosted by a computer server and they are compliant to a single metadata format.

The DLESE Project [16] is very similar to our G-Portal project since both DLESE and G-Portal recognize the importance of tapping content from the Web. DLESE maintains metadata about web entities identified by some user community established by teachers, students, and researchers. These metadata unfortunately must be compliant to a single metadata format that does not include the notion of spatial attributes. Furthermore, DLESE's implementation is not Java based. Hence, complex user interactions cannot be easily supported by DLESE.

#### 1.4. Outline of paper

The remaining sections of this paper present G-Portal in detail. We first give an overview of the system by going through a use case in [Section 2](#). The system architecture is presented in [Section 3](#). [Section 4](#) discusses how resources are represented and organized in G-Portal. [Sections 5](#) and [6](#) cover how resources are visualized and classified, respectively. Finally, we give the concluding remarks and our future work in [Section 7](#).

## 2. Overview of G-Portal

As an educational digital library portal for geography related information, G-Portal must provide a meaningful way to organize its content and services allowing users to effortlessly look for information. In this section, we therefore give an overview of G-Portal by describing a typical example scenario involving a student user. As we walk through the scenario, important terms used in G-Portal and its functional modules will be introduced. More detailed descriptions of these terms and modules will be given in the later sections.

The contents maintained by the G-Portal system are *metadata resources* that describe some pre-existing Web resources located on public domain Web sites, which we call the *raw resources*. For convenience, we shall just refer to the G-Portal metadata resources as *resources* in the rest of our discussion, unless otherwise stated.

In this example scenario, the user is a high school student who would like to prepare for his/her GCE O Level (high school equivalent) geography examination. The student first enters the URL of the G-Portal system in a Web browser and logs on to the system. The system then presents the user with a list of **projects**. Project is a new concept introduced to define the collection of resources that are relevant to a use case or a specific task. In this case, the user selects the project named *GCE Exam* that contains resources related to geography examination. These resources include examination questions, answers, countries, mountains, and other geography resources gathered beforehand by the project creators who are the geography teachers. Within the project, resources are further grouped into **layers** for fine-grained organization as shown in the bottom left window of [Fig. 1](#). As shown in the figure, question resources are grouped into the *Questions* layer. Answers and supplemental materials are grouped into the *Answers & Sup* layer.

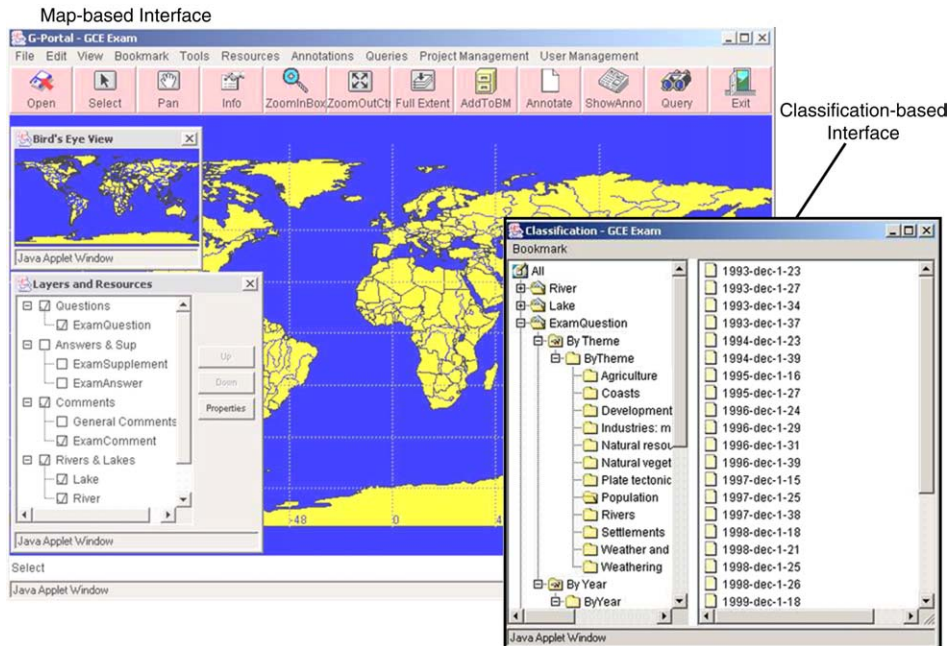


Fig. 1. Map-based and classification-based interfaces.

After the project is loaded, two main user interface windows are shown (see Fig. 1). On the left, resources are drawn as objects on the map in the **map-based interface**. This is a primary point of access to the geospatial resources allowing users to locate a resource with a known location. Usual map navigation aids such as zooming and panning are provided within this map-based interface. Alternatively, a **classification-based interface** is also provided to browse resources classified in taxonomies of categories, especially those resources without spatial features. In this example, since the student is interested in exam questions, which do not have spatial attributes, he/she can begin by browsing examination questions classified by theme or by year using the classification-based interface. It should be noted that the classification-based interface is supported by a flexible **classification scheme** that allows resources to be automatically classified into various taxonomies. For example, examination question resources in the figure have been classified by year and by theme in this project. This is in contrast with the single-taxonomy approach used in almost all the existing digital libraries.

Suppose the user has selected one question from, say, the *population* category. To view the content of the selected resource, the user double-clicks on the question. The content, represented in XML, will appear in an **XML viewer** that we have developed, as shown in Fig. 2(a). Using the XML viewer, the user can view metadata elements nested in XML tags which provide both the semantics and structure meanings of the data. G-Portal represents all its resources in XML and requires resources (including annotation resources) of the same type to be compliant to a **resource schema** that defines the internal structure of

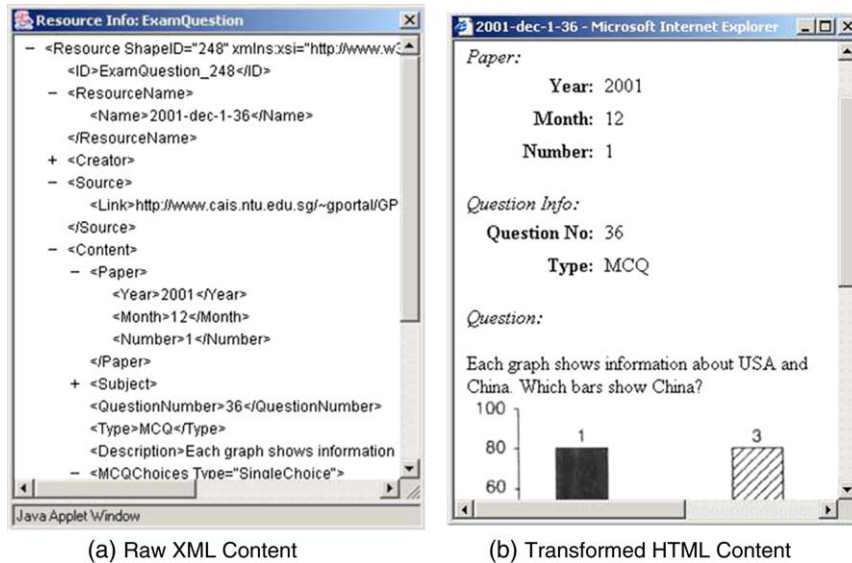


Fig. 2. Visualization of resource content.

the resources. For example, an examination question schema is defined for all examination questions in this project to ensure that all question XML resources provide the same structure. The self-describing and semi-structured nature of XML data model makes it simple to represent a wide range of metadata resources. However, as a high school student, the user may not be familiar with XML. He/she can choose to view the resource content in a more familiar format. In G-Portal, resource contents can also be transformed into HTML web pages using the underlying **resource content transformation** mechanism. Fig. 2(b) shows the HTML counterpart of the resource content in Fig. 2(a). In this example, the transformation involves extracting the URL source of the selected question metadata and returning the raw question resource in HTML using the URL.

While reading the details of the question, the student may notice that the question is about the comparison of information about China and USA. Since countries are also available as resources in the project and are viewable on the map, the student can use the map-based interface to quickly find the two countries and view the contents associated with these two country resources.

Using the population information obtained from these two country resources, the student may arrive at an answer and would like to compare it with the standard answer. Answers in this project are created as **annotations** of the question resources. Resource annotation in G-Portal provides a mechanism to attach additional knowledge to metadata resources and the annotations can be shared. To keep our resource data model simple, we represent annotations as a special type of metadata resources. To view the answer of the selected question, the student finds the answer to the question by asking the system to show the annotations about this question that are of the type *answer*. Like the normal resources, the answer annotation content can be viewed using the XML viewer or Web

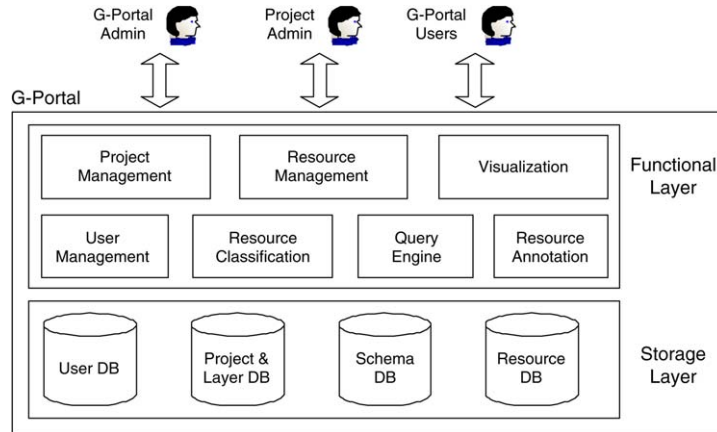


Fig. 3. Design architecture of G-Portal.

browser.<sup>1</sup> The student can then check against their answer and read the detailed explanation in the standard answer.

Having successfully attempted a question, the user may want to contribute a new comment. This can be done by creating an annotation about this country resource with the help of a built-in **XML editor**. Unlike other editors, this XML editor is aware of the schema of the resource (or annotation resource) and is able to guide editing process such that the final resource created will be compliant to the required resource schema.

This concludes our brief overview of G-Portal and a use case example demonstrating how a typical user can use G-Portal to explore, learn, and share knowledge about geography related information.

### 3. System architecture

The overall system architecture of G-Portal is shown in Fig. 3. There are three types of user: **administrator**, **project administrator**, and **users**.

A G-Portal administrator manages all databases in G-Portal using the **project management** and **user management** modules. He/she can create new user accounts, delete existing ones and define users' roles that in turn determine the privileges to access the functional modules in G-Portal. All user information are stored in the **user database**.

Each project administrator uses the project management and **resource management** modules to create and configure projects that can be used by ordinary G-Portal users. The project management module allows project administrators to define projects, create layers that bundle various types of resources, group these layers into the defined projects, and set up appropriate access control. The resource management module lets them register new types of resources (i.e., new resource schemas) and populate the projects with

<sup>1</sup> This requires resource content transformation.



resource instances. The project and layer information are stored in the **project and layer database**. Resource instances and schemas are stored in the **resource database** and **schema database**, respectively. A project administrator can also use the **resource classification** module to associate classification schemas with resources and classify resources.

The rest of G-Portal users with ordinary access rights constitute the majority of targeted users. These users make use of the resource management, **resource annotation**, **query engine**, and resource classification modules to perform various tasks related to geospatial resources, as described in Section 2. The visualization module allows them to access the resources through both map-based and classification-based interfaces and view the resource contents in raw XML format as well as transformed formats such as HTML. The classification module classifies resources on the server and displays the resulting taxonomies in the client's classification interface. Users can query resources based on spatial and non-spatial attributes using the query engine. They can also contribute new annotations about existing resources.

As the focus of this paper is on the implementation issues, we only give brief descriptions of the modules. Due to space constraints, we also exclude the query engine description from the paper. More detailed coverage can be found in [9].

The design architecture shown in Fig. 3 has been implemented in a standard client–server model. The client is a Java applet realizing the visualization module (the map-based and classification-based interfaces) and front-end interfaces of other modules. On the server side is a Java server program that communicates with the client applet and performs necessary operations on the back-end databases to support the front-end interfaces. Therefore, most of the modules depicted in Fig. 3, except the visualization module, have both a client and a server component.

The client and server components are implemented in **client applet** and **server program** respectively. The client applet initially resides on the Web server and is downloaded by the user into the client machine using a Web browser (e.g., Internet Explorer or Netscape Navigator). To support the operations and commands invoked by the user, the applet sends requests to and expects responses from the server program running on the Web server.

The main role of the server program is that of a gateway between the client and the two database servers, namely the **Tamino XML Server** and the **Informix Relational Database Server**. In addition, several modules shown in Fig. 3 (e.g., resource classification and query engine) require part of their operations to be performed in the server program to keep the client applet simple.

## 4. Resource representation and organization

### 4.1. Resource data model

As mentioned earlier, G-Portal focuses specifically on geography related metadata resources and some may be associated with locations on the map. This calls for the explicit inclusion of location in the resource definition. This *location* attribute, together with several other attributes, including *id*, *name*, *owner*, *source*, constitute the core attributes that every



```

<!-- ExamQuestion248.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<Resource xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="ExamQuestion.xsd">
  <ID>ExamQuestion_248</ID>
  <ResourceName> <Name>2001-dec-1-36</Name> </ResourceName>
  <Location>... ..</Location>
  <Creator>... ..</Creator>
  <Source> <Link>
    http://www.cais.ntu.edu.sg/~gportal/GPortal/gce_exam/questions/2001-dec-1-36.htm
  </Link> </Source>
  <Content>
    <Paper> <Year>2001</Year> <Month>12</Month> <Number>1</Number> </Paper>
    <Subject>
      <Title>Each graph shows information about USA ...</Title>
      <Part>Human Geography</Part>
      <Theme>Population</Theme>
      <Topic>Characteristics of population</Topic>
    </Subject>
    <QuestionNumber>36</QuestionNumber>
    <Type>MCQ</Type>
    <Description>Each graph shows information about USA and China.
      Which bars show China?</Description>
    <MCQChoices>
      <Choice> <Label>A</Label> <Description>1, 3, 5</Description> </Choice>
      <Choice> <Label>B</Label> <Description>1, 4, 5</Description> </Choice>
      <Choice> <Label>C</Label> <Description>2, 3, 6</Description> </Choice>
      <Choice> <Label>D</Label> <Description>2, 4, 6</Description> </Choice>
    </MCQChoices>
  </Content>
</Resource>

```

Fig. 4. An example of an ExamQuestion resource.

resource must have. In addition to these mandatory core attributes, each type of resources is allowed to define other new attributes to better describe the resources. The set of mandatory core attributes together with the customized attributes enable the resource data model to represent heterogeneous ranges of metadata. The complete attribute structure shown by resources of the same type is defined by a resource schema. To allow flexible representation of resources with semi-structured attributes, we adopt XML as the resource representation format and XML Schema [17] for defining resource schema.

Fig. 4 depicts the resource of a geography examination question, which also exists as a web page. This resource contains a set of XML elements such as ID, ResourceName, Location, Creator, Source that serve as the core attributes. The XML elements under the Content element are those that are specific to this type of resources, i.e., ExamQuestion resources.

In G-Portal, we allow users to share their knowledge and contribute content to the system by allowing them to create annotations on existing resources [11]. Annotations are represented as a special type of resources, with an additional attribute that locates the annotated resources. Due to space constraints, we shall not elaborate on the annotation scheme further in this paper.

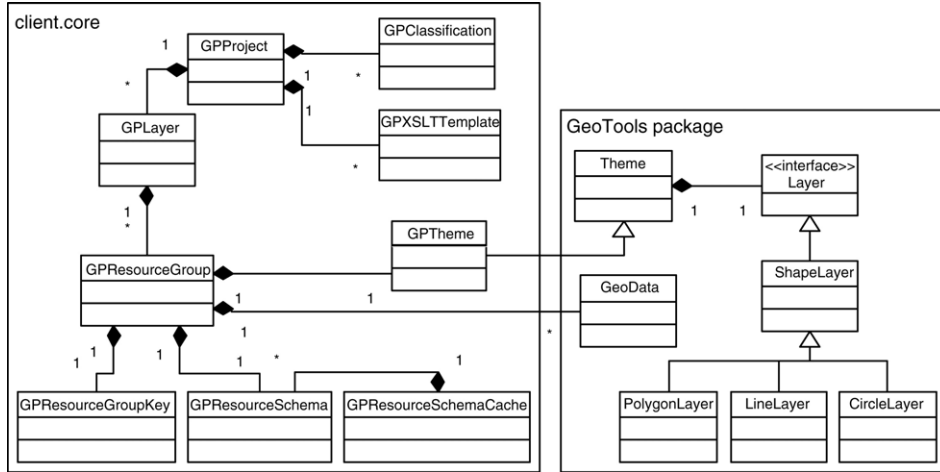


Fig. 5. A class diagram of client side resource representation.

#### 4.2. Resource class design

G-Portal client applet, as described in Section 2, supports several graphical user interface features to visualize the resources in both map-based and classification interfaces. To support the spatially related representation and visualization, we have adopted GeoTools,<sup>2</sup> a public domain GIS package. As it is a general purpose GIS package, the GeoTools' API is too primitive and does not fit well into the resource data model in G-Portal. On the other hand, although XML has been used to represent the resource information, the Java libraries for XML lack the support of GIS features and cannot be directly applied. In G-Portal, we, therefore, adopt a hybrid approach to draw advantages from both the GeoTools package and the Java libraries for XML.

Fig. 5 shows the design of resource related classes. These classes are mainly found in the **client.core** package and the Java packages in the GeoTools. A set of resources of the same type under the same layer are represented by the **GPResourceGroup** class. This class contains a member of the **GPTheme** class, derived from the **Theme** class in GeoTools, that stores the geometry shapes of resources, through the inherited **Layer** interface<sup>3</sup> from **Theme**. The shapes are actually stored in the subclasses that implement the **Layer** interface, depending on the types of the shapes. Other core attributes such as id, name and owner are found in the **GeoData** class, which is also from the GeoTools package. Besides these classes, other utilities from GeoTools, such as those for rendering shapes on the map and for basic map navigation (e.g., zoom-in, zoom-out, and pan), have also been directly incorporated (not shown in Fig. 5).

Instances of the **GPResourceGroupKey** class are used throughout the client applet to maintain the identifiers of **GPResourceGroup** instances each representing a set of

<sup>2</sup> <http://www.geotools.org/>

<sup>3</sup> Note that the term *layer* used in GeoTools is different from the concept of layer in G-Portal.

resources sharing the same resource schema. Each `GResourceGroup` also has a reference to its resource schema in the `GResourceSchema` class. All `GResourceSchema` instances loaded from the server are cached in `GResourceSchemaCache` to avoid reloading the same resource schemas in subsequent project loading.

G-Portal client applet adopts a lazy approach to load resource instances from the server. When the user opens a project, only the location, id, name, and owner of each resource are loaded from the server into the client. The rest of the resource content, including the customized attributes under the `Content` element, are not immediately loaded. This lazy loading strategy achieves better efficiency, as it is unlikely that the user will examine the content of every resource in the loaded project. Instead, it is always preferred to load resource content only upon the user demands, shortening the time required for loading a project. The strategy also works very well for resources that contain large amounts of data. On the other hand, the core attributes such as id, name, and owner are almost always required for several operations, such as display of resource and synchronization between the map-based and classification-based interfaces. Caching these attributes in the client eliminates the need to communicate with the server when performing these operations.

Annotation is implemented in almost the same way as for normal resources, but with an additional boolean attribute to denote whether it is an annotation. Other digital library services provided by the system, such as visualization and classification, will simply treat annotations as resources by ignoring this boolean attribute.

Because the GeoTools package only provides support of general purpose GIS features, we design additional classes to capture the project/layer/resource relationships, as shown in Fig. 5. The `GPPProject` class contains project attributes and a set of instances of `GPLayer` class. `GPLayer` in turn has a set of `GResourceGroup` instances.

#### 4.3. Database design

At the G-Portal server end, persistent storage and efficient queries are the key design issues. The G-Portal server deploys both a Tamino XML Server and an Informix Relational Database Server. G-Portal resources formatted in XML are stored in the Tamino XML server. The XML server supports efficient storage and queries on XML data. As the resource attributes are semi-structured in nature, the XML server can support XPath and XQuery style queries on these resources, which cannot be done using a relational DBMS. In addition to resource contents, resource schemas are also stored in the Tamino server, since they are represented in XML Schemas, which are themselves XML documents.

While the Tamino XML Server can handle the XML style queries on resources, it lacks the support of spatial data types and spatial queries. For example, it is not only cumbersome for the XML server to store shape objects, but also highly inefficient to process spatial queries. To address this deficiency, the Informix Database Server is used. Location attributes are stored in the Informix RDBMS, which provides efficient support of storage and query processing of spatial objects. Since location attributes are always retrieved when loading a project, the Informix database is always accessed. To avoid accessing the other server (the Tamino XML Server), the rest of the core attributes are also stored in the Informix Server so that they can be fetched together with the location attributes. In this way, the speed of project loading is much faster.

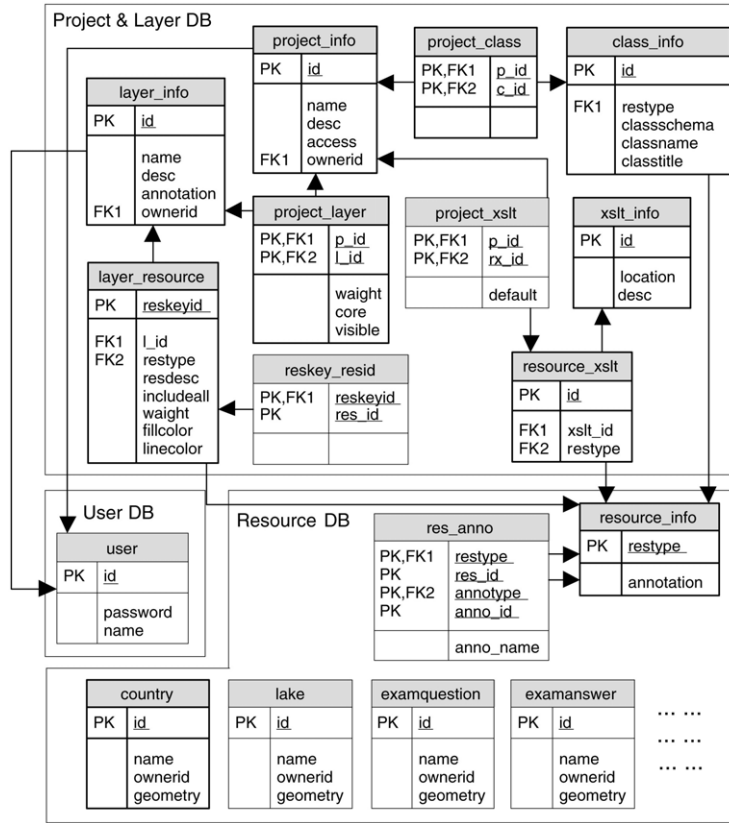


Fig. 6. Database schema design in the Informix RDBMS.

Fig. 6 depicts the database schema design of the Informix RDBMS. In the *resource DB*, each type of resources takes up one table. For example, all country resources are stored in the *country* table and all exam questions in the *examquestion* table. The location attribute is stored in the *geometry* column of each table. The spatial data types of this column depend on the types of resources. For example, we use *multi\_polygon* for countries and *point* for cities. The location attributes are indexed using R-Tree, which provides efficient support for queries involving spatial predicates, such as *coveredby* and *intersect*. Separating resources into individual tables avoids a single large table containing all resources and allows the R-Tree index to adapt itself to more efficiently support the different spatial data types of different resources.

The *resource\_info* table records all the resource types. It also contains another column to indicate whether this type of resource is annotation. There is no distinction between resource and annotation in the way that they are stored in their corresponding tables. The relationships between annotations and the annotated resources are captured in the table *res\_anno*. This enables direct queries involving annotation and the resources that they annotate.

In the *project & layer DB*, the tables `layer_info` and `project_info` store information about layers and projects. The project/layer/resource relationships are determined by the `layer_resource` and `project_layer` tables. All classification schemas reside as individual files on disk and the path to each file is recorded in the `classschema` column of the `class_info` table, along with other attributes of the classification schema. The `project_class` table decides what classification schemas are associated with which project. Similarly, XSLT templates are also stored as files on disk with their location recorded in the `location` column in the `xslt_info` table. Each type of resources can be associated with an XSLT template in the `resource_xslt` table. The `project_xslt` table determines the resource/XSLT relationship to be used in a project and the default XSLT template for that type of resources.

## 5. Resource content visualization and editing

### 5.1. Transformation of XML content

Directly displaying raw XML contents sometimes may not be the preferred way to visualize resource contents because users who are not familiar with XML, especially elementary or high school students, may find it difficult to interpret the nested content of XML resources. To provide users with alternative views of resource contents, we have implemented a mechanism that allows XML contents to be transformed into other formats that are more appealing to users. XSLT has been chosen as the underlying template language for specifying the transformation rules. In the current implementation, we have included templates that transform XML into HTML format, which users are much more familiar with.

XSLT templates are configured by project administrators when defining projects. Each type of resource may have one or more XSLT templates associated with it, representing different ways of transforming the resource content. In a project, for each type of resources, one of the templates is configured (by the project administrator) as the default. Users can choose to view the content in its raw form or after applying the default or selected template. Fig. 2(b) shows the transformed version of the resource in Fig. 2(a) using one of the XSLT templates.

We use the XSLT engine which comes with JDK1.4 to perform the transformation. Since we have to use Java 1.1 in the client, XSLT transformation is performed on the server side and only the transformed content is transferred to the client applet. If the transformed format is HTML, the client applet opens a new Web browser window and writes the HTML content to the window using Javascript codes generated by the client applet. Communication from Java to Javascript is done through LiveConnect<sup>4</sup> with the **netscape.javascript** package, which is implemented in both Netscape and Internet Explorer.

### 5.2. XML editing

To allow users to create and update resources (including annotation resources), it is necessary to have a user-friendly XML editor. Existing commercial or public XML tools

---

<sup>4</sup> <http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/livecon.htm>

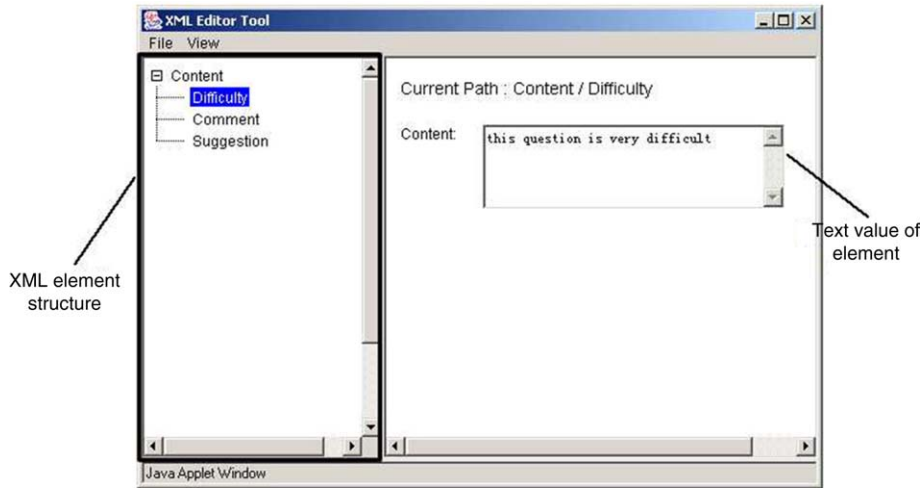


Fig. 7. A screenshot of the XML editor.

do not provide their editing components as independent reusable modules. Our decision with regard to this issue is to implement our own module, called the XML editor. The XML editor is designed to be a component that can either be used within the G-Portal or as an independent tool for other XML editing tasks.

As shown in Fig. 7, the XML editor adopts the familiar Windows Explorer metaphor where the left panel is the XML tree with element names as node names and the right panel displays the content of the selected element. Attribute values and text values of the currently selected element can be entered in the textboxes in the right panel.

The XML editor is schema guided, meaning that it requires a resource schema to guide the editing of an XML resource in terms of its element structure. A schema guided editor has the advantage of being able to create and modify appropriate element structure during the editing process. The editor first parses the XML Schema and generates a base skeleton of the XML content with empty value for each element. The user can then fill in the values of the elements. Insertion and deletion of elements is only allowed when it conforms to the given schema. Before saving the edited XML content, the editor verifies that the filled values satisfy all the constraints in the schema. This procedure not only ensures that the resource content created is valid (conforms to its schema) but also makes the editing process easier for users.

Fig. 8 shows the class design of the XML editor. The design of the `GenericTree` and related classes is a simplified version of that of the `JTree` in the Swing package. The `XMLModel` extends the `TreeModel` base class and overrides the important methods to implement functionalities for parsing XML Schema into a base skeleton and generating XML element instances given the parsed XML Schema. Each node in the skeleton tree is represented by the `XMLModelNode` class, which contains one XML Schema element and one XML element. The classes for dealing with XML Schema and XML documents are

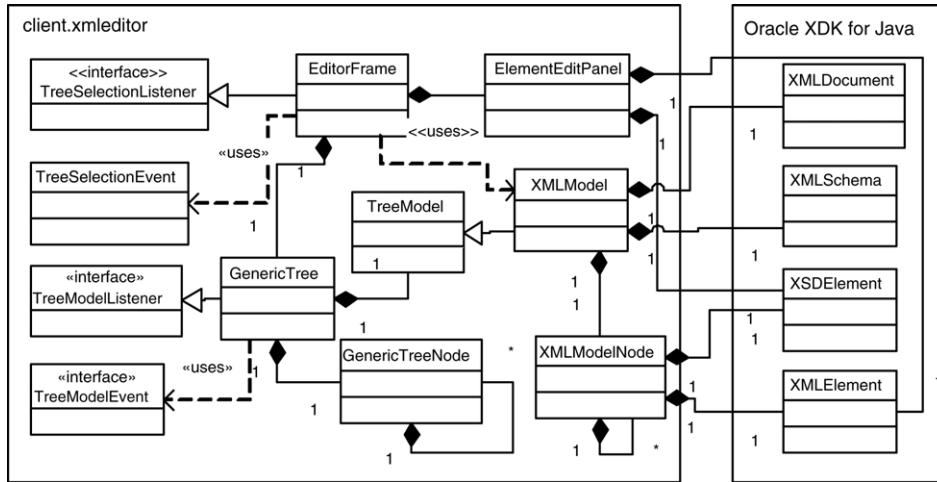


Fig. 8. The class diagram of the XML editor.

borrowed from the Oracle XML Development Kit (XDK) for Java.<sup>5</sup> The Oracle package is chosen because it provides Java classes for representing XML Schema, which most existing Java libraries for XML do not support. The **EditorFrame** instantiates a **GenericTree** object with **XMLModel** and **XMLModelNode** instances. **EditorFrame** also contains a **ElementEditPanel** which is the right hand side panel for editing the content of the current element. The **ElementEditPanel** class receives an **XMLModelNode** instance from the **EditorFrame** whenever the selection in the skeleton tree changes. The XML Schema element in this **XMLModelNode** instance is then used to create labels and textboxes for users to fill in values and attributes of the corresponding XML element instance being edited. The filled values are stored back to the XML element instance in the **XMLModelNode** instance.

## 6. Resource classification

As described in Section 2, G-Portal provides a flexible scheme for classifying the resources according to user-defined classification rules [10]. This section discusses how the classification scheme is implemented in the system.

Resources are classified according to *classification schemas* that dictate how resources are assigned into categories and how the categories are organized into some taxonomy. Each classification schema can be expressed in a specially designed classification language, as illustrated in Fig. 9. In the classification schema shown in Fig. 9, we group the exam question resources by the value of the `/Resource/Content/Subject/Topic` element of each resource and assign them into one of the categories “Natural vegetation”, “Coasts”, “Rivers”, and “OtherTopics”. This classification rule will only be applied to resources

<sup>5</sup> [http://technet.oracle.com/tech/xml/xdk\\_java/content.html](http://technet.oracle.com/tech/xml/xdk_java/content.html)



```

define schema Schema1 on ExamQuestion.xsd;
define rule ByTopicPhysical
  classify by /Resource/Content/Subject/Topic
  grouping {'Vegetation types', 'Human modification of natural vegetation'}
    under 'Natural vegetation',
    {'Coastal processes and resulting landforms', 'Coasts and human activities'}
    under 'Coasts',
    {'Development of a river system', 'River processes and resulting landforms',
      'Rivers and human activities'} under 'Rivers',
  others under OtherTopics
  where /Resource/Content/Subject/Part = 'Physical Geography';

```

Fig. 9. A classification schema for exam questions.

that have a value of “Physical Geography” in their `/Resource/Content/Subject/Part` element. The taxonomy definition portion of this schema is not shown here. Interested readers can refer to [10] for the complete definition of the classification language.

Given a classification schema, the *classification engine* automatically performs the category assignment of resources and constructs the taxonomy accordingly.<sup>6</sup>

The class design of the classification module is shown in Fig. 10. A set of classes (in the top half of Fig. 10, originating from the `ClassificationSchema` class) have been designed to represent the classification schema and its various components. As there could be different actions (e.g., language syntax validation, classification of resources) to be applied to these classes, the *Visitor* pattern [6] is used to keep the set of classes remain stable while allowing new actions to be added. The `ClassificationSchema` and related classes all have one single method `accept()` that takes in an instance of the `Visitor` interface as parameter. Actions such as syntax validation (`ValidationVisitor`), schema conversion (`FromLanguageVisitor`), and resource classification (`ClassificationVisitor`) are implemented as visitor classes inherited from the `Visitor` interface. Passing a different `Visitor` instance to the `accept()` method, a different action can be performed. This eliminates the need for each of the `ClassificationSchema` and related classes to have one method for each action (e.g., `validate()`, `fromlanguage()`, and `classification()`) and makes adding new actions as easy as creating new subclasses implementing the `Visitor` without altering the entire set of classes.

The I/O design allows different sources of input and targets of output repository to be specified. The `CSEngine` class accepts the generic base I/O interfaces. Concrete classes implementing these interfaces can be supplied to the classification engine component. For example, the resources to be classified by the classification engine may be DOM `Element` objects instantiated from the data given by the user using the client applet (`CSElementInput`) or resources directly fetched from the Tamino server (`CSTaminoInput`). In fact, the XML editor introduced in Section 5.2 adopts a similar I/O structure, which allows it to be used within the G-Portal client applet and as a stand-alone tool.

<sup>6</sup> In our implementation, G-Portal uses another *classification schema compiler* to convert a classification schema into its XML equivalent.

server.classification

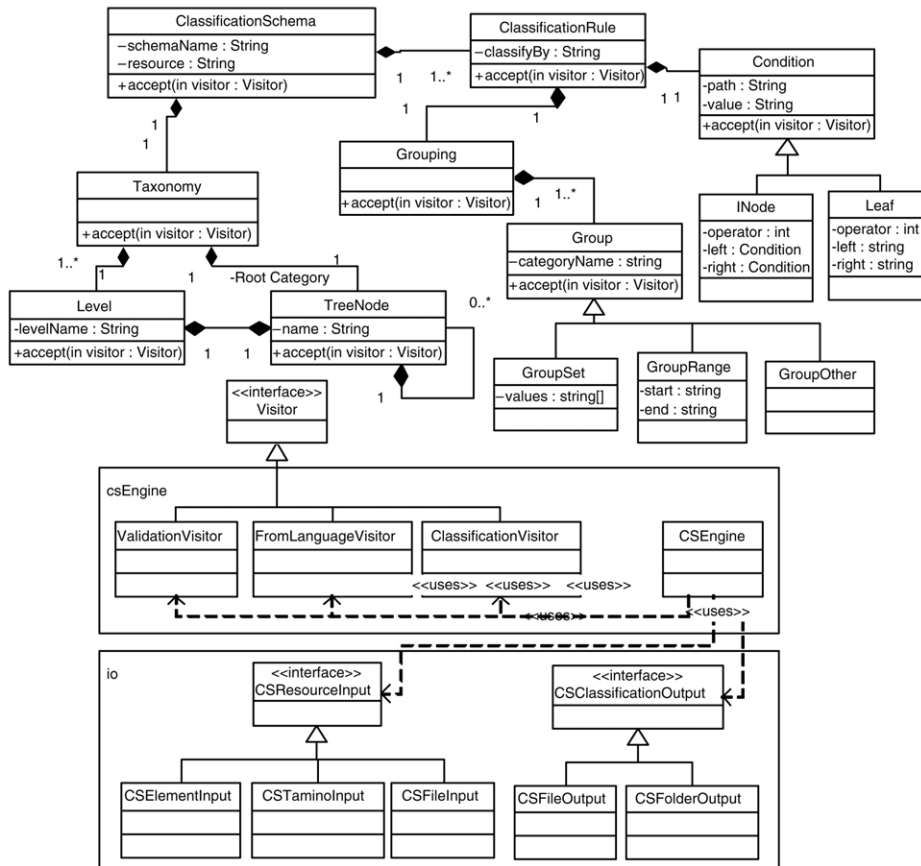


Fig. 10. Design of the classification module.

## 7. Conclusions and future work

In this paper, we give an overview of a unique Java-based Digital Library Portal known as G-Portal. As a geography resource portal, G-Portal offers both a map-based interface and a classification-based interface for organizing and displaying metadata resources that describe public domain Web resources in the geography education domain. The metadata resource representation format adopted by G-Portal is based on XML which is ideal for representing information with flexible structures and for information exchange. We also introduce the concepts of project and layer to organize G-Portal resources for supporting different learning needs.

At present, the G-Portal project has constructed several geography related resource collections including high school geography examination questions, resources harvested from the DLESE project, and country resources. To establish a more systematic approach towards selecting, editing, and sharing quality resources, a G-Portal collection building

team consisting of educators and researchers in the geography domain will be formed in the near future.

As part of our future work, G-Portal will be further extended with more functional modules to support e-learning and community building. Pedagogical studies will be conducted to ensure that G-Portal can support the learning models used in the classroom environment. The notion of personalized project space will also be introduced to support individual students' learning needs. Within a personalized project space, a student can configure a project's content and user interface tools to suit his/her current interests allowing him/her to use the project as a reference in the learning process.

### Acknowledgements

This work is funded by the SingAREN Project M48020004.

### References

- [1] T. Berners-Lee, E. Miller, The semantic web lifts off, *ERCIM News* No. 51 (2002) (October).
- [2] S.K. Cha, K. Kim, C. Song, Y. Kwon, S. Hwang, Efficient web-based access to multiple geographic databases through automatically generated wrappers, in: *Proceedings of the First International Conference on Web Information Systems Engineering, WISE 2000*, IEEE Computer Society Press, Hong Kong, China, 2000, pp. 34–41.
- [3] L.-H. Chua, D.H.-L. Goh, E.-P. Lim, Z. Liu, R.P.-H. Ang, A digital library for geography examination resources, in: *Proceedings of the Second ACM + IEEE Joint Conference on Digital Libraries, JCDL 2002*, Portland, Oregon, USA, 2002, pp. 115–116.
- [4] A. Coleman, T. Smith, O. Buchel, R. Mayer, Learning Spaces in Digital Libraries, in: *Proceedings of Fifth European Conference on Research and Advanced Technology for Digital Libraries, ECDL 2001*, Berlin: Springer, Darmstadt, Germany, 2001, pp. 251–262.
- [5] Columbia Earthscape, <http://www.earthscape.org>, 2002.
- [6] E. Gamma, R. Helm, R. Johnson, J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional Computing Series, Addison-Wesley Publishing Company, New York, NY, 1995.
- [7] F. Lee, S. Bressan, B. Ooi, Hybrid transformation for indexing and searching web documents in the cartographic paradigm, *Information Systems* 26 (2) (2001) 75–92.
- [8] F. Létourneau, Y. Bédard, M.-J. Proulx, SOS-SD A data warehouse-based system for the optimized selection of spatial data, *D-Lib Magazine* (1997) (March).
- [9] E.-P. Lim, D.H.-L. Goh, Z. Liu, W.-K. Ng, C.S.-G. Khoo, S.E. Higgins, G-Portal: A map-based digital library for distributed geospatial and georeferenced resources, in: *Proceedings of the Second ACM + IEEE Joint Conference on Digital Libraries, JCDL 2002*, Portland, Oregon, USA, 2002, pp. 351–358.
- [10] E.-P. Lim, Z. Liu, D.H.-L. Goh, A flexible classification scheme for metadata resources, in: *Proceedings of Digital Library–IT Opportunities and Challenges in the New Millennium, DLOC 2002*, Beijing, China, 2002.
- [11] Z. Liu, E.-P. Lim, D.H.-L. Goh, Resource annotation framework in a georeferenced and geospatial digital library, in: *Proceedings of the 5th International Conference on Asian Digital Libraries, ICADL 2002*, Singapore, 2002, pp. 287–398.
- [12] J. Orendorf, C. Kacmar, A spatial approach to organizing and locating digital libraries and their content, in: *Proceedings of the First ACM Conference on Digital Libraries, DL'96*, ACM Press, Bethesda, MD, USA, 1996, pp. 83–89.
- [13] M.-J. Proulx, Y. Bédard, F. Létourneau, C. Martel, GEOREP: A WWW customizable georeferenced digital library for spatial data, *D-Lib Magazine* (1996) (December).
- [14] T. Smith, A digital library for geographically referenced materials, *IEEE Computer* 29 (5) (1996) 54–60.

- [15] T. Smith, D. Ancona, O. Buchel, M. Freeston, W. Heller, R. Nottrot, T. Tierney, A. Ushakov, The ADEPT concept-based digital learning environment, in: Proceedings of 7th European Conference on Research and Advanced Technology for Digital Libraries, ECDL 2003, Trondheim, Norway, 2003, pp. 300–312.
- [16] T. Sumner, M. Dawe, Looking at digital library usability from a reuse perspective, in: Proceedings of the First ACM + IEEE Joint Conference on Digital Libraries, JCDL 2001, Roanoke, VA, USA, 2001, pp. 416–425.
- [17] XML Schema, <http://www.w3.org/XML/Schema>, 2004.
- [18] B. Zhu, M. Ramsey, H. Chen, R. Hauck, T. Ng, B. Schatz, Create a large-scale digital library for geo-referenced information, in: Proceedings of the Fourth ACM Conference on Digital Libraries, DL 1999, Berkeley, CA, USA, 1999, pp. 260–261.